METHOD FOR STRONGLY ENCRYPTING .ZIP FILES

BACKGROUND OF THE INVENTION

The present invention relates generally to a method of using standard .ZIP files and strong encryption technology to securely store files, and more particularly to a method of integrating existing strong encryption methods into the processing of .ZIP files to provide a highly secure data container which provides flexibility in the use of symmetric and asymmetric encryption technology. The present invention adapts the well established and widely used .ZIP file format to support higher levels of security and multiple methods of data encryption and key management, thereby producing an efficient, highly secure and flexible digital container for electronically storing and transferring confidential data.

Compression of computer files has been available for many years. Compressing files can save large amounts of disk space, and can reduce transfer time when downloading files from the Internet or transferring files through email. Almost any file one downloads from the Internet is compressed in some way. A standard compressed file or folder as it is sometimes called contains one or more files that were compressed into a single file or folder. Many different compression formats have been developed over the years. The .ZIP format, created by the assignee of the present invention, is perhaps the most common compressed file format for the personal computer. Any file with a ".zip" extension most likely contains one or more files of data archived, that is, each either compressed or stored, in the .ZIP format. "Zipping" a file has become a commonly used term meaning to compress the file into the .ZIP format archive so that it occupies less disk space, and similarly, "unzipping" a file means decompressing a compressed file in the .ZIP format.

A .ZIP file is generally recognized as a data compression and archiving format invented by PKWARE, Inc. The .ZIP format is a file format designed for combining data compression technology with file archiving techniques. Many commercially available software products are available for compressing or "zipping" files or other data into the .ZIP format. These .ZIP files can then be used to reconstruct the original data through the "unzipping" process. Data compression converts the contents of a file into an encoded format requiring less computer storage space or in the case of transmission less network bandwidth than the original uncompressed file.

Archiving, in the context of a .ZIP file, is a method of storing information about the characteristics of a file in a catalogue of files, known as the Central Directory, inside the .ZIP file, allowing each file to be retrieved individually by its characteristics. This capability is widely used. These characteristics include, but are not limited to, file name, file size, and file creation date and time.

Software programs such as PKZIP® written by PKWARE, Inc. are used to process files in the .ZIP format. Such programs allow one or more files of any type to be compressed and archived into a file of the .ZIP format type for efficient file storage and transmission over computer and communication networks. This format and the software programs that process .ZIP files have become ubiquitous.

Data encryption is used by many software programs to provide data privacy. Data encryption is a method of encoding data so that it cannot be reproduced in its original form unless an associated key is provided. Decryption uses this key to convert the encrypted data back into its original state. The key is known only to the person encrypting the data or by those other people with whom the person encrypting the data chooses to share the key. The key is used to "unlock" the data so that it can again be used in its original form.

Keys are uniquely generated using data known to the person encrypting a file or other data associated with recipients and users of the file. This data can be a user-defined password or other random data. Several methods are commonly used for processing the keys used for data encryption. Encryption using a key generated from a password is an example of symmetric encryption. Encryption using a public/private key pair is an example of asymmetric encryption. An example of one method for processing encryption keys supported by this invention uses a public/private key pair commonly associated with digital certificates as defined by the document Internet X.509 Public Key Infrastructure Certificate and CRL Profile (RFC 2459). A digital certificate is a unique digital identifier associating a public and private key pair to an assigned individual, a group, or an organization. When used for encrypting data, the public key of an individual is used to process an encryption key which only the individual in possession of the corresponding private key can use for decryption. A digital certificate is issued to an individual, a group, or an organization for a fixed period of time and can only be used during this time period. After the time period has elapsed, the digital certificate will be considered to have expired and must be reissued for a new time period.

The strength of a data encryption method is determined at least in part by its key size in bits. The larger the key size a data encryption method uses, the more resistant it is to cryptanalysis. Cryptanalysis, or popularly "cracking", is the unauthorized access to encrypted data. Strong encryption is a type of data encryption that uses key sizes of 128 bits or more. A number of encryption encoding methods are known today. Examples supported by the present invention include but are not limited to Advanced Encryption Standard (AES), Data Encryption Standard (DES), 2DES, 3DES, and others. A number of key sizes are commonly used today. Examples supported by the present invention include but are not limited to 128 bits, 192 bits, and 256 bits.

Many software programs available today that process .ZIP files use data encryption to encrypt files after compression as they are written to the .ZIP file. The data encryption method used by these software programs uses a key size of 96 bits or less and is considered weak or moderate encryption by today's standards. These software programs use keys generated using user-defined password data. Weak data encryption may not provide sufficient security to computer users that store and transfer their confidential data files using the .ZIP format.

Password-based key generation has been a commonly used method of applying data encryption, however, known vulnerabilities to cracking methods such as "brute force password cracking" make this method of encryption insufficient to meet today's more advanced security needs. Another known limitation of password-based security is the lack of non-repudiation. Non-repudiation is the ability to be certain that the person or program that created an encrypted .ZIP file cannot deny that fact and that their identity is bound to the .ZIP file they created. This cannot be achieved with symmetric encryption methods. Today, non-repudiation is an important aspect of security related to the implementation of digital certificates and digital signatures. It is critically important to be able to prove that a creator or sender of an encrypted file did in fact create the file, i.e. not repudiate his/her action.

Therefore, a need exists to extend the options for levels of security available to programs that process .ZIP files. This extended of security capability makes use of the encryption technologies available today or others that may gain acceptance in the future.

SUMMARY OF THE INVENTION

The present invention provides a method of integrating multiple strong encryption methods into the processing of .ZIP files to provide a highly secure data container which provides flexibility in the use of symmetric and asymmetric encryption technology. The present invention adapts the well established .ZIP file format to support higher levels of security and multiple methods of data encryption and key management, thereby producing a highly secure and flexible digital container for storing and transferring confidential electronic data.

The present invention provides a method of integrating multiple strong encryption methods into the processing of .ZIP files to provide a highly secure data container which provides flexibility in the use of encryption technology. The present invention supports existing weak encryption methods available in .ZIP software programs used today to ensure backward compatibility with existing software programs that use the .ZIP file format. Strong encryption methods are made available to computer users as configurable options to select when compressing and encrypting their files or other data into a .ZIP file.

The method of the present invention provides the capability of using strong encryption when creating .ZIP files. It is flexible in that it provides that different encryption methods can be applied to a single .ZIP file to meet the security needs of a given computer user or application. Strong encryption algorithms are preferably used in conjunction with either password (symmetric) or any form of public/private key (asymmetric) encryption methods. The symmetric method preferably includes a password defined by the user, while the asymmetric method preferably includes a public/private key associated with digital certificates to process encryption keys. The invention allows one or more passwords and one or more public keys to be used individually, or in combination at the same time when archiving any file of any type of data into a secure .ZIP file. This capability is useful since secure .ZIP files are frequently distributed, or otherwise made accessible, to multiple recipients for decryption. Some of those recipients may require password access while others may require certificate access.

The method of the present invention also supports the four basic security functions to be associated with encrypted files: confidentiality, message authentication, sender or creator authentication, and non-repudiation.

Specifically, the present invention supports non-repudiation to uniquely bind a .ZIP file with the identity of its creator, and prevent that creator from denying the creation of that .ZIP file. One method of non-repudiation used by this invention is the identity support available with digital signatures that can be generated using public/private key technology. The non-repudiation function provided by the present invention also preferably supports time-stamping methods for fixing the creation of a digital signature in time, as well as time-stamped audit trails providing transaction history.

As indicated, the method of the present invention also supports message authentication. Message authentication ensures the data has not been altered since being encrypted. The present invention supports message authentication techniques that employ public/private key forms of message authentication, as well as other methods of message authentication that do not require the use of public/private keys. One example of an alternative method that does not use a public/private key is a cryptographic checksum.

The method of the present invention further supports the encryption of file characteristics for each file inside a .ZIP file. Current ZIP software programs encrypt only the contents of the files in a .ZIP file. The additional characteristics for each file, such as its name, size, etc., remain unencrypted. To remove the possibility that this unencrypted data for a file could be made available to an unauthorized user, this information may preferably also be encrypted as an option. This additional encryption further increases the level of security available to .ZIP file users.

Public keys such as those associated with digital certificates used for encrypting .ZIP file data preferably resides on a user's local computer in a file or a database, on an external device such as a Smart Card or other removable device, or in a shared data repository such as a directory service served by an LDAP server.

The present invention also provides multiple methods of checking whether a digital certificate is valid for use. These methods preferably include, but are not limited to standard methods of certificate validation, such as searching certificate revocation lists (CRL), certificate trust lists (CTL), and online checking via the internet using Online Certificate Status Protocol (OCSP) or Simple Certificate Validation Protocol (SCVP).

The method of the present invention also preferably defines data storage locations within the established .ZIP file format specification for storing information on the encryption parameters used when a file was encrypted and on the keys needed when a file is to be decrypted. One such example of these data storage locations includes a field to identify that a new strong encryption method has been applied to a file in the .ZIP file. The strong encryption record will be defined within a Central Directory storage area for each encrypted file. The Central Directory is a storage location defined in the .ZIP file format which serves as a table of contents for the entire .ZIP file. An entry is made into the Central Directory for each file added to a .ZIP file. A decryption record will be defined for storing the information needed to initialize and start the decryption process. This decryption record will be placed immediately ahead of the encrypted data for each file in a .ZIP file. This example is not the only method of storing this data as other storage methods can be defined.

The present invention provides many advantages or benefits over the prior art. One benefit is the ability to use multiple encryption methods instead of supporting only a single encryption method. A second benefit is the ability to use a mixture of symmetric and asymmetric encryption in a single, secure .ZIP file. A third benefit is that the encryption of individual files using advanced public/private keys provides a significantly higher level of security to computer users. A fourth benefit is that encryption of .ZIP file data can be implemented using a range of commonly available cryptographic toolkits. A fifth benefit is that the present invention supports using packaged or readily available encryption algorithms to provide state-of-the-art security. A sixth benefit is the availability of non-repudiation using digital signatures through the use of public/private key technology. A seventh benefit is that the invention ensures a high degree of interoperability and backward compatibility by extending the current .ZIP file format.

Various other features, objects, and advantages of the invention will be made apparent to those skilled in the art from the following detailed description, claims, and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a record layout of a prior art .ZIP file prior to the present invention.

FIG. 2 is a record layout of a .ZIP file in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Referring now to the drawings, FIG. 1 shows the file format for the standard .ZIP file, in existence prior to the present invention. FIG. 2 illustrates the preferred general record layout of a .ZIP file in accordance with the present invention.

The newly modified ZIP file format specification according to the present invention, as published by PKWARE, Inc., is described in a document entitled APPNOTE.TXT, which is attached hereto and incorporated herein by reference. The new version of the .ZIP file format provides an implementation of the use of strong encryption based on a key generated using a password. This implementation constitutes one example of a structure and layout of the records and fields suitable for processing secure .ZIP files as defined by the present invention. The complete description of the conventional or standard .ZIP file format will not be included here since this information is generally well known. Only the portions pertaining to the new records and fields defined by the new format, capable of storing data using strong encryption, will be discussed in detail.

The present invention extends the original .ZIP file format with the addition of new storage records to support the use of strong encryption methods including, as described above, both public/private key, or asymmetric, methods, and password-based, or symmetric, methods, and the capability to use a mixture of symmetric and asymmetric methods.

An example of implementing a new strong encryption method is discussed below. This example identifies several new records and fields that must be defined within the .ZIP file format.

- A new ***General Purpose Bit Flag*** having a hexadecimal value of **0x0040** to be set in both the Local and Central Record Headers when strongly encrypting a file.

- A new ***Decryption Header*** to be located immediately ahead of and adjacent to the compressed data stored for each file.

- A new ***Extra Field*** record definition with an ID having a hexadecimal value of **0x0017** to be inserted into the Central Record Header for each file.

When using these new fields for strongly encrypting files, the following actions are indicated.

1. If the General Purpose Bit Flag value of 0x0040 is set to indicate strong encryption was applied to a file, the General Purpose Bit Flag value of 0x0001 will also generally be set.

2. Files having a size of zero bytes (an empty file) should not generally be encrypted. As indicated, however, the file characteristics of the archived files may be encrypted, even if the file is of zero length and is not itself encrypted.

3. The contents of the field labeled ***Version Needed to Extract*** in both the Local and Central Record Headers should preferably be set to the decimal value of 50 or greater. If the AES

encryption method is used, the contents of the field labeled ***Version Needed to Extract*** in both the Local and Central Record Headers should preferably be set to the decimal value 51 or greater.

4. Data encryption should preferably be applied after a file is compressed, but encryption can be applied to a file if compression is not used. If compression is not applied to a file, it is considered to be **stored** in the .ZIP file.

5. If encryption is applied using digital certificates, a list of intended recipients will be constructed. Each entry in the recipient list identifies a person whose public key has been used in the encryption process for a file and who is allowed to decrypt the file contents using their private key.

Record Definitions:

New Decryption Header (***NDH***)

Value	Size (bytes)	Description
IV size	2	Size of custom initialization vector/salt, if 0 then **CRC32** + 64-bit **File Size** should be used to decrypt data.
IV	variable	Initialization vector/salt (file specific) which should be used in place of **CRC32** + 64-bit **File Size**
Original Size	4	Original (uncompressed) size of the following data
Decryption Info.	variable	***Decryption Information***

Decryption Information (details)

Value	Size (bytes)	Description
Version (3)	2	Version/Format of decryption information.
AlgID	2	Encryption Algorithm ID
BitLen	2	Bit length of the key
Flags	2	Processing flags
ERD size	2	Size of **Encrypted Random Data (ERD)**
ERD	variable	**Encrypted Random Data**
Recipient Count	4	Number of ***Recipients***
Hash Algorithm	2	Hash algorithm to be used to calculate Public Key hash (absent for password based encryption)
Hash Size	2	Size of Public Key hash (absent for password based encryption)
Recipient List Element	variable	***Recipient List Element*** (absent for password based encryption)
Password Validation Data size	2	Size of random password validation data (Includes CRC32 of PVD; >4) MUST be multiple of encryption block sizes
Password, Validation Data	variable	Password Validation Data (PVD)
CRC32 of PVD	4	CRC32 of PVD, used for password verification when decrypting data

Encryption Algorithm ID (AlgID) identifies which of several possible strong encryption algorithms was used for encrypting a file in the .ZIP file. The strong encryption algorithms that can be used include but are not limited to AES, 3DES, 2DES, DES, RC2 and RC4. The use of other unspecified strong algorithms for encryption is supported by the present invention.

Hash Algorithm identifies which of several possible hash algorithms was used for the encryption process for a file in the .ZIP file. The algorithms that can be used include but are not limited to MD5, SHA1 – SHA512. The use of other unspecified algorithms for hashing is supported by the present invention.

Flags

The following values are defined for the processing ***Flags***.

Name	Value	Description
PASSWORD_KEY	0x0001	Password is used
CERTIFICATE_KEY	0x0002	***Recipient List*** is used
COMBO_KEY	0x0003	Either a password or a ***Recipient List*** can be used to decrypt a file
DOUBLE_SEED_KEY	0x0007	Both password and ***Recipient List*** are required to decrypt a file. **ERD** is encrypted twice by 2 separate keys.
DOUBLE_DATA_KEY	0x000f	Both a password and a ***Recipient List*** are required to decrypt a file. File data is encrypted twice using 2 separate keys.
MASTER_KEY_3DES	0x4000	Specifies 3DES algorithm is used for MSK

Recipient List Element

Value	Size (bytes)	Description
Recipient Element size	2	Combined size of Hash of Public Key and Simple Key Blob
Hash	***Hash Size***	Hash of Public Key
Simple key Blob	variable	Simple Key Blob

New Decryption Central Record Extra Field (*NDCEF*)

Value	Size (bytes)	Description
0x0017	2	Signature of ***NDCEF***
Data Size	2	Size of the following data (at least 12 bytes)
Version (2)	2	Version/Format of this extra field.
AlgID	2	Encryption Algorithm ID.
BitLen	2	Bit length of the key
Flags	2	Processing flags
Recipient Count	4	Number of ***Recipients***
Hash Algorithm	2	Hash algorithm to be used to calculate Public Key hash (absent for password based encryption)
Hash Size	2	Size of Public Key hash (absent for password based encryption)
Simplified Recipient List Element	variable	***Simplified Recipient List Element*** (absent for password based encryption)

Simplified Recipient List Element

Value	Size (bytes)	Description
Hash	***Hash Size***	Hash of Public Key

A simplified recipient list element is defined as a subset of a recipient list element and is stored to provide redundancy of the recipient list data for the purposes of data recovery.

Process Flow:

The following is a description of the most preferred encryption/decryption process for a single file using the storage format defined by this example. Any programs, software or other processes available to suitably perform the encryption/decryption process may be used.

Encryption:

1. Validate public/private key
2. Calculate file digital signature and time-stamp
3. Compress or Store uncompressed file data
4. Generate a **File Session Key (FSK)** (see below)
5. Calculate **Decryption Information** size
6. Adjust **Compressed Size** to accommodate **Decryption Information** and padding
7. Save **Decryption Information** to .ZIP file

8. Encrypt **Compressed or Stored File Data**
9. Encrypt file characteristics

Decryption:

1. Decrypt file characteristics
2. Read **Decryption Information** from .ZIP file
3. Generate **FSK** (see below)
4. Verify **Decryption Information** (see below)
5. If Decryption Information is valid, then decrypt **Compressed or Stored File Data**
6. Decompress compressed data
7. Validate file time-stamp and digital signature

Generating **Master Session Key (MSK)**

1. If **MASTER_KEY_3DES** is set, use 3DES 3-key as **MSK** algorithm, otherwise use specified algorithm.
2. If encrypting or decrypting with a password.

2.1.1. Prompt user for *password*

2.1.2. Calculate *hash* of the *password*

2.1.3. Pass calculated *hash* as argument into a cryptographic key derivation function or its equivalent.

3. When encrypting using a public key(s).

3.1.1. Call a cryptographic key generation function or its equivalent to generate random key

4. When decrypting using a private key(s).

    4.1.Using **Recipient List** information, locate private key, which corresponds to one of the public keys used to encrypt **MSK**.

    4.2.Decrypt **MSK**

Salt and/or **Initialization Vector (IV)**

1. For algorithms that use both ***Salt*** and ***IV*,** **Salt** = **IV**
2. **IV** can be completely random data and placed in front of ***Decryption Information***
3. Otherwise **IV** = **CRC32** + 64-bit **File Size**

Adjusting Keys

1. Determine ***Salt*** and/or ***Initialization Vector*** size of the key for the encryption algorithm specified. Usually salt is compliment to 128 bits, so for 40-bit key ***Salt*** size will be 11 bytes. Initialization Vector is usually used by block algorithms and its size corresponds to the block size.
2. If ***Salt*** size > 0 or ***Initialization Vector*** size is > 0 then set **IV**[1] to be used by the specified encryption algorithm.

[1] When adjusting **MSK**, if **IV** is smaller then required Initialization Vector (or Salt) size it is complimented with 0, if it is larger it is truncated. For all other operations **IV** is used as is without any modifications.

Generating **File Session Key (FSK)**

1. **FSK <- SHA1(MSK(IV))**. Adjust **MSK** with **IV**, and decrypt **ERD (Encrypted Random Data)**. Calculate *hash* of **IV + Random Data**. Pass calculated *hash* as argument into a cryptographic key derivation function or its equivalent to obtain **FSK**.

Verifying **Decryption Information**

1. **Decryption Information** contains variable length **Password Validation Data (PVD)**.
2. First **Password Validation Data Size - 4** bytes are random data, and last 4 bytes are CRC32 of that random data. This allows verification that the correct key is used and deters plain text attacks.

The following modifications are used for encrypting and decrypting multiple files.

Multi-file encryption:

1. Generate **MSK**.
2. For each file follow **Encryption** steps.

Multi-file decryption:

1. Generate **MSK** from the file **Decryption Information**
2. For each file follow **Decryption** steps
3. If **Decryption Information** verification fails go to step 1

Alternate storage formats can be defined for implementing the flexible security support within ZIP files. One such alternative is to use other fields, either existing or newly defined to denote that a strong encryption method was applied to a .ZIP archive. Another alternative could be to use additional storage fields in addition to those defined in the above example, or to use the fields as defined, but ordered differently within each record. Still other implementations may use fewer, or more, records or fields than are defined by the above example or the records and fields may be placed in other physical locations within the .ZIP file.

Alternate processing methods can also be defined for implementing the flexible security support within .ZIP files. One such alternative is to implement the encryption process for each file using another public/private key technology such as that defined by the OpenPGP Message Format as documented in RFC 2440. Another alternative could be to use a more direct form of encryption key generation where the file session key is directly used for encrypting each file. This method would not use the indirect form described in the above example where the file session key is derived from a master key.

While the invention has been described with reference to preferred embodiments, it is to be understood that the invention is not intended to be limited to the specific embodiments set forth above. Thus, it is recognized that those skilled in the art will appreciate that certain substitutions, alterations, modifications, and omissions may be made without departing from the spirit or intent of the invention. Accordingly, the foregoing description is meant to be exemplary only, the invention is to be taken as including all reasonable equivalents to the subject matter of the invention, and should not limit the scope of the invention set forth in the following claims.